

[illegible]

```

MM      MM      000000      NN      NN      DDDDDDDD      EEEEEEEEEEE      FFFFFFFFFFF
MM      MM      000000      NN      NN      DDDDDDDD      EEEEEEEEEEE      FFFFFFFFFFF
MMM     MMM     00      00      NN      NN      DD      DD      EE      FF
MMM     MMM     00      00      NN      NN      DD      DD      EE      FF
MM      MM      00      00      NNNN     NN      DD      DD      EE      FF
MM      MM      00      00      NNNN     NN      DD      DD      EE      FF
MM      MM      00      00      NN      NN      DD      DD      EEEEEEEEE      FFFFFFFF
MM      MM      00      00      NN      NN      DD      DD      EEEEEEEEE      FFFFFFFF
MM      MM      00      00      NN      NN      DD      DD      EE      FF
MM      MM      00      00      NN      NN      DD      DD      EE      FF
MM      MM      00      00      NN      NN      DD      DD      EE      FF
MM      MM      00      00      NN      NN      DD      DD      EE      FF
MM      MM      000000      NN      NN      DDDDDDDD      EEEEEEEEEEE      FF
MM      MM      000000      NN      NN      DDDDDDDD      EEEEEEEEEEE      FF

```

```

SSSSSSSS      DDDDDDDD      LL
SSSSSSSS      DDDDDDDD      LL
SS      DD      DD      LL
SS      DD      DD      LL
SS      DD      DD      LL
SS      DD      DD      LL
SSSSSS      DD      DD      LL
SSSSSS      DD      DD      LL
SS      DD      DD      LL
SS      DD      DD      LL
SS      DD      DD      LL
SSSSSSSS      DDDDDDDD      LLLLLLLLLL
SSSSSSSS      DDDDDDDD      LLLLLLLLLL

```

MON
MOD
/*
/*
/*
/*
AGG

CON
END
MOD
/*
/*
/*
/*
AGG

CON
END

MODULE \$cddbdef;

/* Class Descriptor Block

```
/*
/**
/*
/* Data structures for Monitor utility
/*
/*--
/*
/*      Version 'V04-000'
/*
/******
/*
/* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
/* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
/* ALL RIGHTS RESERVED.
/*
/* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
/* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
/* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
/* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
/* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
/* TRANSFERRED.
/*
/* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
/* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
/* CORPORATION.
/*
/* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
/* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
/*
/******
/*
/***
/* FACILITY:  MONITOR Utility
/*
/* ABSTRACT:  Data Structure Definitions
/*
/* ENVIRONMENT: Non-executable.
/*
/* AUTHOR:   Thomas L. Cafarella, April, 1981
/*
```

```
/*
/* MODIFIED BY:
/*
/* V03-017 TLC1090 Thomas L. Cafarella 02-Aug-1984 15:00
/* Correct ACCVIOs in SYSTEM and PROCESSES classes.
/*
/* V03-016 TLC1087 Thomas L. Cafarella 25-Jul-1984 15:00
/* Default to /ALL when summarizing.
/*
/* V03-015 TLC1085 Thomas L. Cafarella 22-Jul-1984 14:00
/* Calculate scale values for Free and Modified List bar graphs.
/*
/* V03-014 TLC1072 Thomas L. Cafarella 17-Apr-1984 11:00
/* Add volume name to DISK display.
/*
/* V03-013 TLC1066 Thomas L. Cafarella 01-Apr-1984 11:00
/* Add SYSTEM class.
/*
/* V03-012 TLC1060 Thomas L. Cafarella 19-MAR-1984 10:00
/* Make multi-file summary work for homogeneous classes.
/*
/* V03-011 PRS1011 Paul R. Senn 29-Feb-1984 14:00
/* add /FLUSH_INTERVAL qualifier
/*
/* V03-010 PRS1006 Paul R. Senn 17-FEB-1984 14:00
/* Add support for "computed" items
/*
/* V03-010 TLC1052 Thomas L. Cafarella 17-Feb-1984 11:00
/* Add multi-file summary capability.
/*
/* V03-009 PRS1005 Paul R. Senn 13-JAN-1983 10:00
/* Add display control field to CDB and CHD to
/* allow flexible spacing between screen items
/*
/* V03-008 TLC1051 Thomas L. Cafarella 11-Jan-1984 11:00
/* Add consecutive number to class header record.
/*
/* V03-008 PRS1001 Paul R. Senn 27-Dec-1983 16:00
/* Add ALL classes flag to MRB
/*
/* V03-007 TLC1050 Thomas L. Cafarella 06-Dec-1983 11:00
/* Change directory information in DLOCK class.
/*
/* V03-006 TLC1048 Thomas L. Cafarella 11-Jun-1983 12:00
/* Remove UIC from PROCESSES displays.
/*
/* V03-005 TLC1042 Thomas L. Cafarella 19-Jun-1983 15:00
/* Add /ITEM qualifier for homogeneous classes.
/*
/* V03-005 TLC1039 Thomas L. Cafarella 15-Jun-1983 15:00
/* Add DECnet node name to heading.
/*
/* V03-005 TLC1036 Thomas L. Cafarella 10-Jun-1983 15:00
/* Properly recognize Revision Level 0.
/*
```

```
/*      V03-004 TLC1035      Thomas L. Cafarella      06-Jun-1983      15:00
/*      Add homogeneous class type and DISK class.
/*
/*      V03-003 TLC1028      Thomas L. Cafarella      14-Apr-1983      16:00
/*      Add interactive user interface.
/*
/*      V03-003 TLC1027      Thomas L. Cafarella      14-Apr-1983      16:00
/*      Enhance file compatibility features.
/*
/*      V03-001 TLC0014      Thomas L. Cafarella      01-Apr-1982      13:00
/*      Correct attached processor time reporting for MODES.
/*
/*      V03-002 TLC1012      Thomas L. Cafarella      30-Mar-1982      13:00
/*      Display user's comment string on screen line 5.
/*
/*      V03-001 TLC1009      Thomas L. Cafarella      29-Mar-1982      01:00
/*      Get current time when other times are converted.
/*
/*--
/*
```

```

/*
/* Define constants used to define display types. These codes will be stored
/* in the CDB$B_ST field in the Class Descriptor Block (CDB).
/*

```

```

CONSTANT (
    reg_proc,      /* Code for regular PROCESSES display
    topc_proc,     /* Code for TOPCPU PROCESSES display
    topd_proc,     /* Code for TOPDIO PROCESSES display
    topb_proc,     /* Code for TOPBIO PROCESSES display
    topf_proc      /* Code for TOPFAULT PROCESSES display

```

```

    ) EQUALS 0 INCREMENT 1 COUNTER #procdisp ;

```

```

CONSTANT procdisp EQUALS #procdisp+1 ;

```

```

CONSTANT (
    all_stat,      /* Code for ALL statistics
    cur_stat,      /* Code for CURRENT statistic
    ave_stat,      /* Code for AVERAGE statistic
    min_stat,      /* Code for MINIMUM statistic
    max_stat       /* Code for MAXIMUM statistic

```

```

    ) EQUALS 0 INCREMENT 1 COUNTER #stats ;

```

```

CONSTANT stats EQUALS #stats+1 ;

```

```

/*
/* Define Class Descriptor Block offsets. There is a Class Descriptor
/* Block for each class of performance data.
/*

```

```

AGGREGATE cdb STRUCTURE PREFIX cdb$ ; /* Class Descriptor Block
    faostr      LONGWORD;      /* Length of FAO control string
    faostr      ADDRESS;       /* Address of FAO control string
    sumbuf      LONGWORD;      /* Length of multi-file summary buffer
    sumbuf      ADDRESS;       /* Address of multi-file summary buffer
    title       ADDRESS;       /* Address of title cstring
    icount      LONGWORD;      /* Number of items in this class (STD)
                                /* Number of TOP items in PROCESSES class (non-STD)
    ecount      LONGWORD;      /* Number of display elements in this class
    itmstr      ADDRESS;       /* Address of item token string (STD)
                                /* Address of PROCESSES Display Descriptor (non-STD)
    blklen      WORD;          /* Length of a block
    precoll     ADDRESS;       /* Address of pre-collection routine (0 if none)
    postcoll    ADDRESS;       /* Address of post-collection routine (0 if none)
    buffers     LONGWORD;      /* Length of collection buffer block
    buffers     ADDRESS;       /* Address of block of collection buffers (STD)
                                /* Address of block consisting of collection buffer
                                /* and display buffer (non-STD)
    cdx         ADDRESS;       /* Address of CDB extension for homog class (0 if not)
    dispctl     WORD;          /* Display control bit string
    min         LONGWORD;      /* Minimum value
    range       LONGWORD;      /* Value range (max-min)

```

```

faoseglen    BYTE;          /* Length of FAO segment for data display (used by homogs)
faoprelen    BYTE;          /* Length of FAO prefix for data display (used by homogs)
st           BYTE;          /* Code for form of statistical display (Active)
st_def       BYTE;          /* Default ST code
st_cur       BYTE;          /* Current ST code
qflags       STRUCTURE TAG w; /* Class qualifier flags for CDB (Active)
percent      BITFIELD LENGTH 1 MASK; /* Percent = YES => User has requested all data be displayed
                                         as percent values.
cpu          BITFIELD LENGTH 1 MASK; /* Cpu = YES => User has requested MODES to be displayed
                                         in CPU-specific format
gfiller      BITFIELD LENGTH 16-^; /* Fill out remainder of word
END qflags;
qflags_def   WORD;          /* Default class qualifier flags
qflags_cur   WORD;          /* Current class qualifier flags
flags        STRUCTURE TAG l; /* Flags for cdb
ctpres       BITFIELD LENGTH 1 MASK; /* YES => this class has at least one count item
swapbuf      BITFIELD LENGTH 1 MASK; /* Swap buffers bit is flip-flopped
uniform      BITFIELD LENGTH 1 MASK; /* Uniform can be YES or NO
                                         Uniform = YES => This class is strictly made up of items
                                         which can be expressed as percentages of
                                         a whole.
cpu_comb     BITFIELD LENGTH 1 MASK; /* YES => combine collected items for display
std          BITFIELD LENGTH 1 MASK; /* YES => this is a standard class
homog        BITFIELD LENGTH 1 MASK; /* YES => this standard class is homogeneous
diskac       BITFIELD LENGTH 1 MASK; /* YES => this is the disk class with allocation class names
diskvn       BITFIELD LENGTH 1 MASK; /* YES => this is the disk class with volume names
syscls       BITFIELD LENGTH 1 MASK; /* YES => this is the special SYSTEM class
disable      BITFIELD LENGTH 1 MASK; /* YES => Do not allow this class to be requested
kunits       BITFIELD LENGTH 1 MASK; /* YES => Bar graph headings displayed in K units
wide         BITFIELD LENGTH 1 MASK; /* YES => Special wide screen display used by DISK
explic       BITFIELD LENGTH 1 MASK; /* YES => Class qualifier specified explicitly
filler       BITFIELD LENGTH 32-^; /* Fill out remainder of longword
END flags;
chhdr        ADDRESS;       /* Address of CHange Descriptors header
#cdbsize = .; /* Size of cdb
END cdb;

```

```

CONSTANT size EQUALS #cdbsize PREFIX cdb$ ; /* Declare constant for cdb size

```

```

END_MODULE $cdbdef;

```

MODULE \$cdxdef;

/* Class Descriptor Block Extension

```
/*
/* This structure is an extension to the CDB for
/* homogeneous classes.
/*
```

AGGREGATE cdb_ext

STRUCTURE PREFIX cdx\$; /* CDB Extension

```
  ibits          BITFIELD LENGTH 16 TAG w; /* Active item bits. If a bit is set, the item
/* ... with that bit number has been requested
  ibits_def      WORD:                      /* Default item bits. See above.
  ibits_cur      WORD:                      /* Current item bits. See above.
  idiscf         BYTE:                      /* Number of items requested for display
  idisconsec     BYTE:                      /* Consecutive number of current display item
  idisindex      BYTE:                      /* Item index of current display item
  elidlen        BYTE:                      /* Length of an element ID
  cumelct        WORD:                      /* Cumulative element count for this MONITOR request
  elidtable      ADDRESS:                   /* Element ID Table address
  scbtable       ADDRESS:                   /* STATS Control Block Table address
  selidtable     LONGWORD:                  /* Super Element ID Table length
  selidtable     ADDRESS:                   /* Super Element ID Table address
  dcount         LONGWORD:                  /* Count of elements to display this time
  prev_dct       LONGWORD:                  /* Count of elements displayed last time
  ilooktab       ADDRESS:                   /* Address of item keyword lookup table
  dispnam        ADDRESS:                   /* Address of rtn to display element names
  dispfao        ADDRESS:                   /* Address of FAO control string for elt names
  #cdxsize = .; /* Size of CDX
END cdb_ext;
```

CONSTANT size

EQUALS #cdxsize PREFIX cdx\$; /* Constant for CDX size

END_MODULE \$cdxdef;

```
MODULE $chddef;                                /* Change Descriptor
/*
/* Define Change Descriptor (CHD) offsets. There is one CHD for each change
/* to the item structure of a class (and one for the original state).
/* The CHDs for a given class are contiguous and immediately follow the
/* CHD Header. The CHD Header is a single byte containing the current
/* Revision Level for the class. Following the CHD Header is a number
/* of CHDs equal to the Revision Level + 1. Revision Level 0 represents
/* the original state of the item structure.
/* The CDB contains a pointer to the CHD Header. Each CHD defines a new
/* Revision Level for the class. A class with one CHD is at Rev Level 0,
/* a class with two CHDs is at Rev Level 1, etc.
/*

AGGREGATE chd  STRUCTURE PREFIX chd$ ;          /* Change Descriptor
    icount      LONGWORD;                      /* Number of items in this class (STD)
    itmstr      ADDRESS;                       /* Number of items for TOPs (PROCESSES class) (non-STD)
    blklen      WORD;                         /* Address of item token string (STD)
    elidlen     BYTE;                         /* Address of PDD (PROCESSES class) (non-STD)
    dispctl     WORD;                         /* Block length (STD HETEROGENEOUS)
    #chdsize = .;                             /* Data block length (PROCESSES class & STD HOMOGENEOUS)
    END chd;                                  /* Element ID length (STD HOMOGENEOUS)
                                           /* display control bit string
                                           /* Size of CHD

CONSTANT size  EQUALS #chdsize PREFIX chd$ ; /* Constant for CHD size

END_MODULE $chddef;
```

MODULE \$idbdef;

/* Item Descriptor Block

```

/*
/* Define Item Descriptor Block (IDB) offsets. There is one Item Descriptor
/* Block for each unique data item. Generally, a data item is defined for
/* only one class, although there are some instances of data items which
/* are defined for several classes (Page Fault Rate, for example).
/*

```

```

AGGREGATE idb      STRUCTURE PREFIX idb$ ;      /* Item Descriptor Block
sname              ADDRESS;                     /* Address of short name cstring
lname              ADDRESS;                     /* Address of long name cstring
isize              WORD;                        /* Code indicating size of data item
type              WORD;                        /* Code indicating type of data item
/* NOTE -- Size and type codes are defined
/*              in module MONDAT.MAR
addr              ADDRESS;                     /* Address of data item (initialized by
/*              BLDIDB macros in module MONDAT.MAR)
flags              STRUCTURE TAG b;             /* Flags for IDB
pcnt              BITFIELD LENGTH 1 MASK;       /* YES => computed percentage item
filler            BITFIELD LENGTH 8-^;         /* Fill out remainder of byte
END flags;
#idbsize = .;
END idb;

```

```

CONSTANT ilength   EQUALS #idbsize PREFIX idb$ ; /* Constant for IDB size

```

END_MODULE \$idbdef;

MODULE \$mrdbef;

/* Monitor Request Block

```

/*
/* Define Monitor Request Block (MRB) offsets. There is one Monitor Request
/* Block for each monitor request. A monitor request is defined as
/* one MONITOR subcommand invocation.
/*

```

AGGREGATE mrb STRUCTURE PREFIX mrb\$;

/* Monitor Request Block

```

beginning QUADWORD; /* Beginning time of request in system time units
ending QUADWORD; /* Ending time of request in system time units
interval LONGWORD; /* Interval value in seconds
flush LONGWORD; /* Flush interval in seconds
viewing_time LONGWORD; /* Viewing time for a screen in seconds
input ADDRESS; /* Address of input file descr (0 if input not requested)
display ADDRESS; /* Address of display file descr (0 if display not requested)
record ADDRESS; /* Address of record file descriptor (0 if record not requested)
summary ADDRESS; /* Address of summary file descriptor (0 if summary not requested)
comment ADDRESS; /* Address of comment string descriptor
classct WORD; /* Count of classes requested
classbits OCTAWORD UNSIGNED; /* Bit string of requested classes
inp_files BYTE; /* Count of input files specified
flags STRUCTURE TAG w; /* Flags for MRB
display BITFIELD LENGTH 1 MASK; /* YES => user requested /DISPLAY
record BITFIELD LENGTH 1 MASK; /* YES => user requested /RECORD
summary BITFIELD LENGTH 1 MASK; /* YES => user requested /SUMMARY
playback BITFIELD LENGTH 1 MASK; /* YES => user requested /INPUT
indefend BITFIELD LENGTH 1 MASK; /* YES => ending time is indefinite
disp_to_file BITFIELD LENGTH 1 MASK; /* YES => user specified a filename on /DISPLAY
inp_cl_req BITFIELD LENGTH 1 MASK; /* YES => input cleanup required
rec_cl_req BITFIELD LENGTH 1 MASK; /* YES => record cleanup required
dis_cl_req BITFIELD LENGTH 1 MASK; /* YES => display cleanup required
sum_cl_req BITFIELD LENGTH 1 MASK; /* YES => summary cleanup required
all_class BITFIELD LENGTH 1 MASK; /* YES => ALL classes requested
mfsum BITFIELD LENGTH 1 MASK; /* YES => multi-file summary requested
by_node BITFIELD LENGTH 1 MASK; /* YES => m.f. summary by node requested
syscls BITFIELD LENGTH 1 MASK; /* YES => SYSTEM class is being monitored
proc_req BITFIELD LENGTH 1 MASK; /* YES => PROCESSES class explicitly requested
filler BITFIELD LENGTH 16-*; /* Fill out rest of word
END flags;
#mrbsize = .;
END mrb;

```

/* Size of mrb

CONSTANT size

EQUALS #mrbsize PREFIX mrb\$; /* Constant for mrb size

END_MODULE \$mrdbef;

MODULE \$mca\$;

/* Monitor Communication Area

```

/*
/* This "structure" consists of unrelated variables used by the various
/* routines of the MONITOR utility. They have been placed in a based
/* structure for ease of reference across separately compiled PL/I
/* and MACRO-32 modules.
/*

```

AGGREGATE mca STRUCTURE PREFIX mca\$;

```

input_len      LONGWORD; /* Monitor Communication Area
input_ptr      ADDRESS;  /* Length of current input file record
intticks      LONGWORD; /* Address of current input file record
collcnt       LONGWORD; /* Interval calculation (in 10ms ticks)
dispcnt       LONGWORD; /* Count of collections completed
int_mult      LONGWORD; /* Count of displays completed
               /* Interval multiple (for playback, # of intervals
               /* ... to advance before recording or displaying)
proc_disp     LONGWORD; /* Number of processes to display (PROCESSES class)
mpaddr        ADDRESS;  /* Address of MP (multiprocessing) code
curr_time     QUADWORD; /* Current time in system time units
lastcoll      QUADWORD; /* Time stamp of latest collection
firstc       BYTE;      /* Class number of first requested class
lastc        BYTE;      /* Class number of last requested class
flags         STRUCTURE TAG w; /* Flags
"entry"      BITFIELD LENGTH 1; /* Type of entry -- can be COMMAND or UTILITY
future       BITFIELD LENGTH 1; /* YES => monitor request begins in future
multfnd      BITFIELD LENGTH 1; /* Multiple found can be YES or NO
eof          BITFIELD LENGTH 1; /* YES => EOF (end-of-file) on /INPUT file
video        BITFIELD LENGTH 1; /* YES => Display terminal is a video device
graphics     BITFIELD LENGTH 1; /* YES => Display terminal is a VT55
era_scri     BITFIELD LENGTH 1; /* YES => PROCESSES scrolling region must be erased
top_disp     BITFIELD LENGTH 1; /* YES => At least one TOP display event has occurred
refresh      BITFIELD LENGTH 1; /* YES => Screen refresh request received (CTRL-R, CTRL-W)
s_top_disp   BITFIELD LENGTH 1; /* YES => At least one SYSTEM (top) display event has occurred
filler       BITFIELD LENGTH 16-^; /* Fill out rest of word

END flags;
consec_rec    LONGWORD; /* Consecutive number for recorded collection events
dclassct     WORD;      /* Count of requested classes being displayed
#mcasize = .; /* Size of MCA
END mca;

```

CONSTANT size

EQUALS #mcasize PREFIX mca\$; /* Constant for MCA size

END_MODULE \$mca\$;

```
MODULE $mbpdef;                                /* Monitor Buffer Pointers
```

```
/* This structure consists of ten pointers to MONITOR collection and
/* statistics buffers. The pointers themselves are located at the
/* beginning of a block of space consisting of the pointers followed
/* immediately by the buffers.
```

```
/* For the non-standard class PROCESSES (regular display), there are
/* only 3 buffers:
```

```
/*      Buffera, which is the collection buffer,
/*      Buff1st, which is the 1st collection buffer, and
/*      Pr_faostk, which is the display (FAO stack) buffer.
```

```
/* For the non-standard class PROCESSES (TOP display), there are the
/* three buffers above, plus 5 buffers used to do TOP calculations.
```

```
AGGREGATE mbp STRUCTURE PREFIX mbp$ ;          /* Monitor Buffer Pointers
      buffera ADDRESS;                          /* Pointer to collection buffer A
      bufferb ADDRESS;                          /* Pointer to collection buffer B
/* The above two pointers may not be moved !!
      stats ADDRESS;                            /* Pointer to statistics buffer
      min ADDRESS;                             /* Pointer to buffer containing min values
      max ADDRESS;                             /* Pointer to buffer containing max values
      sum ADDRESS;                             /* Pointer to sum buffer
      pcstats ADDRESS;                         /* Pointer to percent statistics buffer
      pcmin ADDRESS;                          /* Pointer to buff with min percent values
      pcmax ADDRESS;                          /* Pointer to buff with max percent values
      pcsum ADDRESS;                          /* Pointer to percent sum buffer
      #mbpsize = .;                            /* Size of MBP
END mbp;
```

```
CONSTANT size EQUALS #mbpsize PREFIX mbp$ ; /* Constant for MBP size
```

```
AGGREGATE mbp2 STRUCTURE PREFIX mbp$ ;         /* Monitor Buffer Pointers for PROCESSES/TOP class
      buffa ADDRESS;                           /* Pointer to collection buffer A
      buff1st ADDRESS;                        /* Pointer to 1st collection buffer of MONITOR request
      data ADDRESS;                          /* Pointer to DATA array
      diff ADDRESS;                          /* Pointer to DIFF array
      order ADDRESS;                         /* Pointer to ORDER array
      pid ADDRESS;                           /* Pointer to PID array
      addr ADDRESS;                          /* Pointer to ADDR array
END mbp2;
```

```
AGGREGATE mbp3 STRUCTURE PREFIX mbp$ ;         /* Monitor Buffer Pointers for PROCESSES (REG) class
      ba ADDRESS;                             /* Pointer to collection buffer A
      b1st ADDRESS;                          /* Pointer to 1st collection buffer of MONITOR request
      pr_faostk ADDRESS;                     /* Pointer to PROCESSES FAO stack
END mbp3;
```

```
END_MODULE $mbpdef;
```

MODULE \$scbdef;

/* STATS Control Block

```
/*
/* This structure maintains information about corresponding elements
/* in the Element ID Table. The Element ID Table contains information
/* about each element (e.g., each disk) in a homogeneous class (e.g., DISK).
/* The information is a string of characters for each element which
/* serve to identify the element.
/*
```

AGGREGATE stats_block STRUCTURE PREFIX scb\$; /* STATS Control Block

```
    dbidx      WORD;          /* Data block index into PREVIOUS collection buffer
    flags      STRUCTURE TAG b; /* Flags
    current     BITFIELD LENGTH 1; /* The corresponding element in the Element ID Table
                                     /* ... was collected during the most recent interval
    active      BITFIELD LENGTH 1; /* The corresponding element in the Element ID Table
                                     /* ... is actively being collected
    filler      BITFIELD LENGTH 8-^; /* Fill out rest of byte
    END flags;
    #scbsize = .; /* Size of SCB
    END stats_block;
```

CONSTANT size EQUALS #scbsize PREFIX scb\$; /* Constant for SCB size

END_MODULE \$scbdef;

MODULE \$tm1def; /* Temporary storage for FILL_HOMOG_STATS

/*
/* This structure consists of definitions for temporary storage
/* used by the FILL_HOMOG_STATS routine.
/*

AGGREGATE temp_1_block STRUCTURE PREFIX tmp\$; /* Monitor Buffer Pointers

dbct	LONGWORD;	/* Data block count
dblen	LONGWORD;	/* Data block length
elidct	LONGWORD;	/* Number of element ID table elements
dbidx	LONGWORD;	/* Data block index
found	BYTE;	/* "element found" indicator
#tm1size = .;		/* Size of TM1
END temp_1_block;		

CONSTANT size EQUALS #tm1size PREFIX tmp\$; /* Constant for TM1 size

END_MODULE \$tm1def;

MODULE \$tm2def; /* Temporary storage for FILL_MFSUM_FAOSTK

/*
/* This structure consists of definitions for temporary storage
/* used by the FILL_MFSUM_FAOSTK routine.
/*

AGGREGATE temp_2_block STRUCTURE PREFIX tm2\$; /* Temporary storage

start_col	LONGWORD;	/* Starting column number
cols	LONGWORD;	/* Number of columns to display in summary report
col_size	LONGWORD;	/* Number of longwords in a column
cols_used	LONGWORD;	/* Number of columns used
seconds	LONGWORD;	/* Number of seconds
colls	LONGWORD;	/* Number of collections
elems	LONGWORD;	/* Count of elements to display for current class
item_type	WORD;	/* Item type
#tm2size = .;		/* Size of TM2
END temp_2_block;		

CONSTANT size EQUALS #tm2size PREFIX tm2\$; /* Constant for TM2 size

END_MODULE \$tm2def;

```
AGGREGATE temp_3_block STRUCTURE PREFIX tm3$ ; /* Temporary storage
```

```

inpidx      LONGWORD;      /* Index into elt ID table for curr elt
supidx      LONGWORD;      /* Index into Super elt id table for curr elt
items       LONGWORD;      /* Number of items requested
sbccl       ADDRESS;       /* Addr of portion of summary buffer for curr column
sblen       LONGWORD;      /* Length of portion of summary buffer for one item
found       BYTE;          /* Element Found indicator (lbs means "found")
#tm3size =  /* Size of TM3
END temp_3_block;

```

```
CONSTANT size EQUALS #tm3size PREFIX tm3$ ; /* Constant for TM3 size
```

```
MODULE $stm4def;                                /* Temporary storage for FILL_DISP_BUFF */
/*
/* This structure consists of definitions for temporary storage
/* used by the FILL_DISP_BUFF routine.
/*
```

```
AGGREGATE temp_4_block STRUCTURE PREFIX tm4$ ; /* Temporary storage
```

```

ecount          LONGWORD;          /* Count of elements to display
itmstr          ADDRESS;           /* Address of display item string
buffers         ADDRESS;           /* Address of MBP (buffer block)
fltsecs         LONGWORD;          /* Floating point count of seconds
#tm4size =      /* Size of TH4
END temp_4_block;

```

```
CONSTANT size           EQUALS #tm4size PREFIX tm4$ ; /* Constant for TM4 size
```

```
END_MODULE Stm4def;
```

```
MODULE $ifbdef;                                     /* Input File Block
/*
/* This structure contains information about an input file
/* used for a multi-file summary.
/*
```

```

AGGREGATE ifb    STRUCTURE PREFIX ifb$ BASED ifbptr; /* Input File Block
    input        ADDRESS; /* Address of input filename string descriptor
    col no       BYTE;    /* Column no. (1-origin) which includes data from this file
    #ifbsize = .; /* Size of Input File Block
END ifb;

```

```
CONSTANT size      EQUALS #ifbsize PREFIX ifb$ : /* Constant for IFB size
```

COL
V04

0

```
END_MODULE $ifbdef;
```

```
MODULE $csbdef;
```

```
/* Column Summary Block
```

```
/*  
/* This structure contains information about a single column  
/* of data in a multi-file summary report.  
/*
```

```
AGGREGATE csb STRUCTURE PREFIX csb$ BASED csbptr; /* Column Summary Block  
  nodename CHARACTER LENGTH 16; /* Node name of data source node for this column  
  beginning QUADWORD; /* Beginning time of this column's data in system time units  
  ending QUADWORD; /* Ending time of this column's data in system time units  
  flags STRUCTURE TAG b; /* Flags  
  ignore BITFIELD LENGTH 1; /* YES => ignore this column  
  filler BITFIELD LENGTH 8-^; /* Fill out rest of byte  
  END flags;  
  files BYTE; /* Number of files included in this column  
  #csbsize = .; /* Size of Column Summary Block  
  END csb;
```

```
CONSTANT size EQUALS #csbsize PREFIX csb$ ; /* Constant for CSB size
```

```
END_MODULE $csbdef;
```

MODULE \$mfsdef;

/* Multi-File Summary Block

```
/*  
/* This structure contains information about a Multi-File  
/* Summary request. There is one MFS per MONITOR request.  
/*
```

```
AGGREGATE mfs  STRUCTURE PREFIX mfs$ BASED mfsptr: /* Multi-File Summary Block  
  classbits    OCTAWORD UNSIGNED; /* Bit string of requested classes  
  beginning    QUADWORD; /* Requested beginning time  
  ending       QUADWORD; /* Requested ending time  
  statsbuf     LONGWORD; /* Length of statistics buffer (for TOT,MIN,MAX)  
  statsbuf     ADDRESS; /* Address of statistics buffer (for TOT,MIN,MAX)  
  ifb_tab      ADDRESS; /* Address of IFB Table  
  summary      ADDRESS; /* Address of summary file descriptor  
  elems        LONGWORD; /* Count of elements to display for current class  
  lwords       LONGWORD; /* Count of longwords on FAOSTK for 1 element  
  classct      WORD; /* Count of requested classes  
  columns      BYTE; /* Count of columns in summary report  
  cur_col      BYTE; /* Number of column currently being processed  
  data_cols    BYTE; /* Number of columns with data (as opposed to blank)  
  #mfssize = .; /* Size of Multi-File Summary Block  
END mfs;
```

```
CONSTANT size      EQUALS #mfssize PREFIX mfs$ ; /* Constant for MFS size
```

END_MODULE \$mfsdef;

MODULE \$mondef;

/* Monitor Recording File Definitions

```

/*
/* These definitions describe data items in MONITOR Recording File records.
/* The record types include: recording file header record, system information
/* record, class header portion of class record and prefix portion of
/* PROCESSES class record.
/*

```

```

/* NOTE -- The recording file header record and the system information record
/*           require that offset symbols be defined for any fields added after
/*           the initial release. The definition of the symbol appears as a
/*           CONSTANT just before the field definition itself. Then, in any code
/*           references to the field, the offset should be compared against the
/*           the actual size of the record to determine whether the field is
/*           present in the record. This technique allows any MONITOR image
/*           to process any recording file, regardless of its structure level.
/*

```

```

#maxcomlen = 60;
#faosize = 4*16;

```

```

/* Max length of user comment string
/* Number of bytes for FAO stack (display buffer)
/*   for a single process (PROCESSES class)

```

```

AGGREGATE file_hdr
type          STRUCTURE PREFIX mnr_hdr$ : /* Monitor File Header Record
flags         BITFIELD LENGTH 8 TAG b; /* Unsigned record type
filler       STRUCTURE TAG 1; /* Flags
END flags;    BITFIELD LENGTH 32-^; /* Fill out rest of longword
beginning    QUADWORD; /* Beginning time of request in system time units
ending       QUADWORD; /* Ending time of request in system time units
interval     LONGWORD; /* Interval value in seconds
rev0clsbits  OCTAWORD UNSIGNED; /* Bit string of recorded classes which are at rev 0
/* NOTE -- The above item is included for compatibility with
/*           MONITOR structure levels MONSL001 and MONBA001
recct        LONGWORD; /* Count of all records in the file (incl header)
level        CHARACTER LENGTH 8; /* MONITOR Recording File structure level identification
comment      CHARACTER LENGTH #maxcomlen; /* User comment string
comlen       WORD; /* Actual length of user comment string
classbits    CONSTANT classbits EQUALS .; /* Bit string of recorded classes
revlevels    CONSTANT revlevels EQUALS .;
#flhsz = .;  CHARACTER LENGTH 128; /* Rev level for each recorded class
END file_hdr; /* Size of file header

```

```

CONSTANT size          EQUALS #flhsz PREFIX mnr_hdr$ : /* Constant for file header size
CONSTANT maxcomlen     EQUALS #maxcomlen PREFIX mnr_hdr$ : /* Constant for user comment string size

```

```

AGGREGATE sys_info
type          STRUCTURE PREFIX mnr_sysi$ : /* Monitor System Information Record
/* Contains VAX/VMS system info about the monitored system
flags         BITFIELD LENGTH 8 TAG b; /* Unsigned record type
clusermem     STRUCTURE TAG w; /* Flags
reserved1     BITFIELD LENGTH 1; /* YES => this node is a member of a cluster
/* Reserved to DIGITAL
/* For MON08001 (V4FT1), was used as DIRNODE, where
/* YES => this node is a lock mgr directory node
filler        BITFIELD LENGTH 16-^; /* Fill out rest of word

```

```

END flags;
boottime      QUADWORD;          /* Absolute system boot time in system time units
maxprcct      WORD;              /* MAXPROCESSCNT SYSGEN parameter
mpcpus        BYTE;              /* Number of multiprocessing CPUs

CONSTANT nodename EQUALS ;
nodename      CHARACTER LENGTH 16; /* DECnet node name of data source node
CONSTANT balsetmem EQUALS ;
balsetmem     LONGWORD;          /* Balance set memory (in pages)
CONSTANT mpwhilim EQUALS ;
mpwhilim      LONGWORD;          /* Modified Page List high limit (in pages)
CONSTANT cputype EQUALS ;
cputype       LONGWORD;          /* CPU type code (in binary)
#sysize = .;   /* Size of sys info record
END sys_info;

CONSTANT size      EQUALS #sysize PREFIX mnr_sys$ ; /* Constant for sys info record size

AGGREGATE class_hdr STRUCTURE PREFIX mnr_cls$ ; /* Monitor Class Record Header
type               BYTE;          /* Unsigned record type
flags              STRUCTURE TAG w; /* Flags
cont               BITFIELD LENGTH 1; /* The data for this interval continues in next record
filler             BITFIELD LENGTH 16-^; /* Fill out rest of word
END flags;
stamp              QUADWORD;       /* System time of collection
reserved           WORD;           /* Reserved to DIGITAL
#clhsize = .;      /* Size of class header
END class_hdr;

CONSTANT hsize     EQUALS #clhsize PREFIX mnr_cls$ ; /* Constant for class header size

AGGREGATE hom_class_pre STRUCTURE PREFIX mnr_hom$ ; /* Homogeneous Class Prefix
eltct              LONGWORD;       /* Count of elements in this record
reserved           LONGWORD;       /* Reserved to DIGITAL
#hpfsz = .;        /* Size of homog class prefix
END hom_class_pre;

CONSTANT psz       EQUALS #hpfsz PREFIX mnr_hom$ ; /* Constant for homog class prefix size

AGGREGATE pro_class_pre STRUCTURE PREFIX mnr_pro$ ; /* PROCESSES Class Prefix
pctrec             LONGWORD;       /* Count of processes in this record
pctint             LONGWORD;       /* Count of processes for this interval
#ppfsz = .;        /* Size of PROCESSES prefix
END pro_class_pre;

CONSTANT psz       EQUALS #ppfsz PREFIX mnr_pro$ ; /* Constant for PROCESSES prefix size

/*
/* PROCESSES Class Data Block. This structure defines the offsets for items in the
/* data block for the (non-standard) PROCESSES class. New items may be added only
/* to the end of the existing structure. Furthermore, any item item annexed must
/* be checked for its presence when processing an input recording file. That is,
/* compare the offset value for the new item to the size of the data block being
/* examined; if the offset value is less than the data block size, the item is
/* present in the input file, otherwise, it is not. Whenever an item is added, a
/* new Revision Level is defined for the PROCESSES class. Insert a symbol defining
/* the boundary of the revision level, and add a CHD macro in MONDAT.MAR referencing

```

```
/* the boundary symbol (which effectively defines the new data block size).
/*
```

```
AGGREGATE process_class STRUCTURE PREFIX mnr_pro$ : /* PROCESSES Class Data Block
    ipid      LONGWORD; /* Internal PID
    uic       LONGWORD; /* UIC (Member is low-order word)
    state     WORD; /* State value
    pri       BYTE; /* Priority (negative value)
    lname     OCTAWORD; /* Process name (counted string)
    gpgcnt    WORD; /* Global page count
    ppgcnt    WORD; /* Process page count
    sts       LONGWORD; /* PCB Status Vector
                /* (PCBSV RES bit clear => swapped out)
    diocnt    LONGWORD; /* Direct I/O count
    pageflts  LONGWORD; /* Page fault count
    cputim    LONGWORD; /* Accumulated CPU time (in ticks)
    bioct     LONGWORD; /* Buffered I/O count
    CONSTANT rev0dsize EQUALS . ; /* Revision Level 0 boundary

    epid      LONGWORD; /* Extended PID
    efwm      LONGWORD; /* Event flag wait mask (for MWAITS)
    CONSTANT rev1dsize EQUALS . ; /* Revision Level 1 boundary

    #pdbname = .; /* Size of process data block
END process_class;
```

```
CONSTANT dsize EQUALS #pdbname PREFIX mnr_pro$ : /* Constant for process data block size
CONSTANT fsize EQUALS #faosize PREFIX mnr_pro$ : /* Constant for FAOSTK (display buffer) size
```

```
AGGREGATE qualifier_desc STRUCTURE PREFIX qual$ BASED qualptr; /* Qualifier Descriptors
    beg      LONGWORD; /* /BEGINNING qualifier length
    beg      ADDRESS; /* /BEGINNING qualifier address
    "end"     LONGWORD; /* /ENDING qualifier length
    "end"     ADDRESS; /* /ENDING qualifier address
    int      LONGWORD; /* /INTERVAL qualifier length
    int      ADDRESS; /* /INTERVAL qualifier address
    flush    LONGWORD; /* /FLUSH_INTERVAL qualifier length
    flush    ADDRESS; /* /FLUSH_INTERVAL qualifier address
    view     LONGWORD; /* /VIEWING_TIME qualifier length
    view     ADDRESS; /* /VIEWING_TIME qualifier address
    inp      LONGWORD; /* /INPUT qualifier length
    inp      ADDRESS; /* /INPUT qualifier address
    disp     LONGWORD; /* /DISPLAY qualifier length
    disp     ADDRESS; /* /DISPLAY qualifier address
    rec      LONGWORD; /* /RECORD qualifier length
    rec      ADDRESS; /* /RECORD qualifier address
    summ     LONGWORD; /* /SUMMARY qualifier length
    summ     ADDRESS; /* /SUMMARY qualifier address
    comm     LONGWORD; /* /COMMENT qualifier length
    comm     ADDRESS; /* /COMMENT qualifier address
    by_node  LONGWORD; /* /BY_NODE qualifier length
    by_node  ADDRESS; /* /BY_NODE qualifier address
    class    LONGWORD; /* /CLASS_NAME parameter length
    class    ADDRESS; /* /CLASS_NAME parameter address
    all      LONGWORD; /* /ALL qualifier length
    all      ADDRESS; /* /ALL qualifier address
```

```
cur      LONGWORD;      /* /CURRENT qualifier length
cur      ADDRESS;       /* /CURRENT qualifier address
ave      LONGWORD;      /* /AVERAGE qualifier length
ave      ADDRESS;       /* /AVERAGE qualifier address
min      LONGWORD;      /* /MINIMUM qualifier length
min      ADDRESS;       /* /MINIMUM qualifier address
max      LONGWORD;      /* /MAXIMUM qualifier length
max      ADDRESS;       /* /MAXIMUM qualifier address
topc     LONGWORD;      /* /TOPCPU qualifier length
topc     ADDRESS;       /* /TOPCPU qualifier address
topd     LONGWORD;      /* /TOPDIO qualifier length
topd     ADDRESS;       /* /TOPDIO qualifier address
topb     LONGWORD;      /* /TOPBIO qualifier length
topb     ADDRESS;       /* /TOPBIO qualifier address
topf     LONGWORD;      /* /TOPFAULT qualifier length
topf     ADDRESS;       /* /TOPFAULT qualifier address
cpu      LONGWORD;      /* /CPU qualifier length
cpu      ADDRESS;       /* /CPU qualifier address
pcent    LONGWORD;      /* /PERCENT qualifier length
pcent    ADDRESS;       /* /PERCENT qualifier address
item     LONGWORD;      /* /ITEM qualifier length
item     ADDRESS;       /* /ITEM qualifier address
#qdsiz = .;
END qualifier_desc;

AGGREGATE def_desc      STRUCTURE PREFIX def$ BASED defptr; /* Default Qualifier Value Descriptors
  rec      LONGWORD;    /* /RECORD qualifier default value length
  rec      ADDRESS;     /* /RECORD qualifier default value address
  disp     LONGWORD;    /* /DISPLAY qualifier default value length
  disp     ADDRESS;     /* /DISPLAY qualifier default value address
  summ     LONGWORD;    /* /SUMMARY qualifier default value length
  summ     ADDRESS;     /* /SUMMARY qualifier default value address
  #ddsize = .;
  END def_desc;

END_MODULE $mondef;
```

0239

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY